Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 22 (2013) 643 – 652
Procedia Computer Science

17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013

Automation of message handling in cloud-based managed service

Daisuke Yamada, Yoshitaka Kuwata, Ryosei Kasai, and Tatsuya Nakamura
NTT DATA CORPORATION, Tokyo, 135-6033, Japan

Abstract

Abstract—As the scale of the computer systems becomes very large and complex in the latest cloud data centers, the operations of these systems should be more efficient. We analyzed the operations of an actual data center and found that approximately 30% of the time was spent on system message handling. We proposed a concept of an automated message handling system for a cloud-based managed service. This system is based on modeling an operator's actions in a state transition model. We built a prototype system and evaluated its performance for three months on an actual data center. We observed that approximately 80% of the message handling was automated with the system.

Keywords -Runbook automation;data center operation; system monitoring

1. Introduction

In recent years, new categories of services called "cloud services" have become more popular among major information service providers. The cloud service includes the following three categories.

1) IaaS (Infrastructure as a Service) provides servers and networks using virtualization technology.
2) PaaS (Platform as a Service) provides middleware for applications.
3) SaaS (Service as a Service) provides applications to the end users.

We call these types of services a "cloud data center". In IaaS, systems are virtualized; a tenant system is composed of a set of virtual servers on which applications are deployed. Most of the IaaS cloud services are designed to be used as "self service" models. Tenants of IaaS rent the servers, launch them, and deploy systems on top of them on their own.

In contrast, the "managed service" model is often used to outsource the operation of a system before the emergence of a cloud service. In managed service, operations of tenant systems are conducted by the operators of managed service providers (MSP). The managed service model is still used for a cloud data center. The larger the scale that the cloud data center is, the more systems the MSP must handle.

doi:10.1016/j.procs.2013.09.145

1.1. Managed Services

Cloud data center that provides managed services must take care of both the operation of their tenant systems and the common system infrastructure. According to the progress of virtualization technologies, the more the systems are integrated into cloud data centers, the more the systems must be run simultaneously by a provider of managed services. As an operators' workload tends to increase, more efficient work is required.

To order the managed services, the service level (SL) must be defined and agreed on between the tenant and the MSP beforehand. The SL for the tenant system includes policies, rules, work procedures, and reports. Documents that describe the detailed operation procedures are called the "runbook". They describe work procedures, including how to make a backup, monitor messages, and handle the messages. Operators' work for managed services is conducted based on runbooks.

1.2. Analysis of Managed Services in Practice

As the operators' workload affects the efficiency of the operations in managed services, the definition of the runbooks is very important. We picked up one typical managed service and analyzed the operators' workload. We found that the operators spend approximately 30% of their work time on the monitoring of system messages. In the monitoring work, the operator checks all of the incoming messages. When the operator receives a message, for example, he or she refers to the runbook to find a corresponding message and decides the necessary actions to take for the message. If correspondence actions are required for the message, then the operator performs the actions.

To improve the efficiency of the managed services, it is necessary to improve message handling with the runbook. In this paper, we propose a system that automates the handling of messages. We propose a model of message handling that can be applied to various managed services.

2. Operations of Managed Services

2.1. Lifecycle Model of Managed Services

The following is a lifecycle model of managed services. There are four phases in the model.

- system design phase
- system deployment phase
- system operation phase
- system removal phase

When the operation of a system is outsourced to MSP, a runbook is created in the system design phase. The main operation of the managed services is performed by the runbook in the system operation phase.

We classify the operation of managed services into three categories: the scheduled work, the unscheduled work, and the monitoring work. The table below shows an example of a task for each operation.

Table 1. Operation of managed services

Operation	Example of task
Scheduled work	Visual inspection of hardware
	Exchange magnetic tapes and cleaning
	Backup data
	Update anti-virus check pattern
	Make reports
Unscheduled work	Start / stop servers
	Start / stop services

Operation	Example of task
	Check log messages
	Restore data
Monitoring work	Handle system messages
	Activate / deactivate monitoring system

2.2. Runbook

A runbook is created for each system, in which specific instructions on the tasks are described. In general, a runbook includes the following documents.

- System design: This document describes the design of systems, such as the hardware configuration and the network configuration.
- Operation tasks: This document describes the work procedures of scheduled tasks and unscheduled tasks.
- Message handling tasks: This document describes the procedure of message handling. This procedure is called the "runbook rule".
- Schedule: This document describes the schedule of operations.

Operators perform the system operation according to the schedule. When operators receive requests from the tenant, the operators update the schedule and perform the task, which is called an unscheduled task. When operators receive messages from monitoring systems, the operators perform the tasks according to the runbook rule. Table 2 shows an example of runbook rules.

To increase the efficiency of a managed service, operators must increase the efficiency of the message handling tasks because they are the most time-consuming.

Table 2. An example of runbook rules

Runbook rule	Description
Repeat rule	When operators receive a message, the operators must hold the message until the following message. Then, the operators determine the action from combinations of messages.
Threshold rule	If the value in a certain time period has continued to exceed the threshold, the operator determines that work is required. For instantaneous thresholds that are exceeded, resource monitoring is not a problem.
Ignore result rule	If this rule is applied, then the message is abandoned immediately. Operators are not required to perform actions. This rule is often used for systems in the maintenance phase.

Simple string matching methods have been commonly used for message handling systems. When the operators receive two or more messages, they should consider the combination and order of these messages. However, string matching methods cannot be applied in these cases. According to runbook rules, operators must estimate the status of systems from the sequences of their messages. Therefore, we propose an automated message handling system with a state transition model to represent the runbook rules.

3. Related Studies on the Management of Data Centers

Greenberg et al. [2] performed a cost analysis of hardware among cloud-like data centers, which includes servers, infrastructure, power requirements, and networking. In addition to these costs, we focus on the operational costs in this paper.

Michael Isard [3] reported an automated data center management system. This system is focused on automating software provisioning and deployment. In this paper, we focus on the operations phase and attempt to minimize the associated cost.

Yamamoto et al. [4] proposed an automated message system that is based on a stream database. In their framework, a sequence of messages is considered a 'complex event'. Techniques from Complex Event Processing (CEP) are heavily used for their research. In contrast, we applied a knowledge-based approach in our framework, which is based on the model of operations used by actual managed services.

Several open source software systems for log management are very popular and are widely used for the management of servers. This category of software includes logsurfer [1], logsurfer [5] and swatch [6].

4. State Transition Model for Automated Message Handling

To implement an automated message handling system, we defined a set of state transition models. Each of these models represents the model of an operator's recognition and decision. In this section, we describe the state transition models in detail.

4.1. Overview of the Automated Message Handling System

The automated message handling system receives messages from monitoring systems. The system applies runbook rules to the messages and sends only the messages that the operators need to know. The system consists of three modules, a message receiver, a message analyzer, and a message sender, as shown in Figure 1.

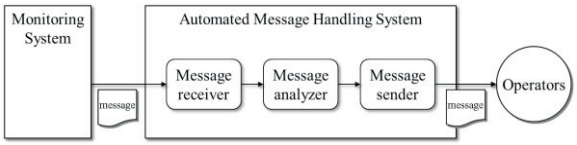

Figure 1. Architecture of the automated message handling system

- Message receiver: This module receives messages from monitoring systems, normalizes the messages into standard format, and sends them to a message analyzer.
- Message analyzer: To decide whether the messages requires corresponding actions or not, this module applies a runbook rule to incoming messages. If required, the message is passed on to the message sender.
- Message sender: This module sends messages to human operators.

4.2. State Transition Model

According to the runbook, the operators must estimate the status of the systems from the sequences of their messages. The estimation rules can be modeled as several state transition models, with the following terms.

- STATE: State represents the current status of the system, as estimated from the sequence of messages.
- TRANSITION: Transition represents a change of a state to another state.
- ACTION: Action represents an action required at a transition.
- TIMER: When a certain amount of time passes, the state changes to another state.
- COUNTER: Counter keeps track of the number of states in the model.

We implemented three models, which correspond to the three rules shown in Table 2.

4.2.1. Repeat Model

This model corresponds to a runbook rule called “repeat rule”. Figure 2(a) represents a state diagram for the repeat model.

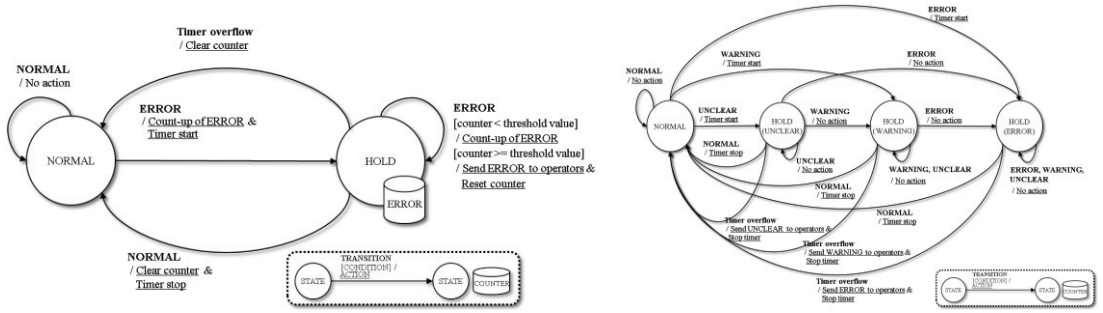

Figure 2. (a) State diagram for the repeat model; (b) State diagram for the threshold model

This model keeps checking the state in a certain interval. An occasional ERROR status might be observed in the messages because of the fraction of the system. However, if a normal message is observed in the next check, the system will be back in a NORMAL state, and no action is required by human operators.

In this state diagram, when an ERROR state is observed in the sequence of the messages, the state changes to HOLD and a timer is started. If a normal state is observed in the next check, or if the timer has overflowed, then the error message is abandoned and the state changes to NORMAL. As a result, the operators must take care of the error message. If the ERROR state is continuously observed in the message, and the counter reports errors more than certain threshold values, then the error message is sent to the operator the first time. At this point, operators should start working on the error message.

4.2.2. Threshold Model

This model corresponds to a runbook rule called the “threshold rule”. Figure 2(b) represents a state diagram for the threshold model.

An occasional error/warning/unclear status might be observed in the messages. However, if the normal state is observed within a certain period of time, it will be judged as normal and no actions are taken by operators. If the error/warning/unclear status is continuously observed within a certain period of time, then the error/warning/unclear message is sent to the operator.

4.2.3. Ignore Result Model

This model corresponds to a runbook rule called “ignore result rule”. Any message that matches this rule is abandoned, and no action is performed by human operators. This model is used to ignore messages from systems regarding maintenance.

4.3. Design of the Rule Description Language

These state transition models are based on the rule definitions in the runbook. A rule is composed of two sections. The first section is called the “common section” and represents the pre-conditions of the rule. When a pre-condition is matched to a message, the rule is applied to the message. The second section includes the threshold value, which determines the trigger of a state change within a model. A detailed rule description for each model is shown in Table 3.

The rules are described in YAML form. Figure 3 shows an example of a repeat model, which contains two rules called repeat01 and repeat99. Rule repeat01 is applied to messages that contain a string "MailSystem."

This rule estimates that the state is NORMAL if the message contains "OK". Similarly, this rule estimates that a state is ERROR if the message contains "NG".

```
- rulename : repeat01
  rule : MESSAGE==‘MailSystem.*'
 success_msg : $MESSAGE==‘.*OK.*'
 failure_msg : $MESSAGE==‘.*NG.*'
 time : 60
 count : 3
- rulename : repeat99
 rule : MESSAGE==‘IaaSSystem.*'
 success_msg : $MESSAGE==‘.*Deploy.*'
 failure_msg : $MESSAGE==‘.*Failure.*'
 time : 180
  count : 5
```

Figure 3. Example of a rule description for *repeat model*

Table 3. Rule description language

Section	Item	Description
Common	rule name	The identifier of a rule
	rule	If the messages match regular expressions, then this rule is applied to the messages.
Repeat model (second section)	success_msg	If the message matches a regular expression, then the state is estimated as NORMAL.
	failure_msg	If the message matches the regular expression, then the state is estimated as ERROR.
	time	Time of repeat.
	count	Number of repeat times.
Threshold model (second section)	time	Time to hold a message.
	start	The start date of this rule. The date is described in cron format.
	duration	Duration of this rule.
Ignore result model (second section)	start	The start date of this rule. The date is described in cron format.
	duration	Duration of this rule.

5. Evaluation

5.1. System Architecture for a Prototype System

To evaluate the usability of the state-transition models of the runbook, we built a prototype of an automated message handling system. Figure 4 represents the system architecture of the prototype system.

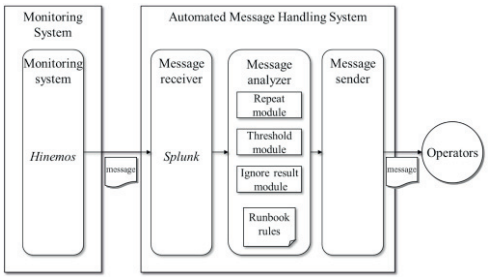

Figure 4. System architecture of the prototype system

- Monitoring system: Hinemos [7] is used to monitor the state of the nodes. It also monitors the capacity of the resources, such as the CPU, memory, and disks, that are used by the systems.
- Message receiver: Splunk [8] is used to handle incoming messages. Because each system has a unique format, Splunk normalizes these messages into a standard format.
- Analytic component: This component is a core component of the system, and it includes a state transition model. This component keeps track of the states of the nodes, holds messages, deletes unnecessary messages, and sends the remaining messages to operators to take actions. This component is written in Python.
- Message sender: This component sends messages to operators. This component is written in Python.

5.2. Evaluation

To evaluate the reduction in the workload for operators, we applied our prototype system to our actual data center. We measured the work hours as a metric of the adaptability of the model and the effect of automation in the message handling.

Table 4 shows the overview of the data center that we used for the evaluation.

Note that the number of nodes changed in the evaluation term. This change occurred because some systems extended their servers. With regard to the construction in the system, the monitoring system tends to receive more messages than usual.

Table 4. Overview of the data center

ITEM	NUMBER	NOTE
Systems	30	Number of commercial systems
Nodes	600	Number of servers
Monitoring System	35	Number of monitoring systems, including Hinemos servers
Message	50~150 / day	Number of messages received by the prototype system

In the data center, an IaaS platform is used, and all of the servers are virtualized. Each of the systems is composed of several virtualized servers and a set of monitoring systems. The monitoring system is designed as an echo virtualized system. These systems are responsible for the monitoring of systems. The messages from all of the monitoring systems will be collected by an automated message handling system and will be sent to system operators. Messages include the status of the nodes, the monitoring of processes in the nodes, log monitoring from the nodes, and service port monitoring.

Figure 5 shows the data flow model of the message handling system.

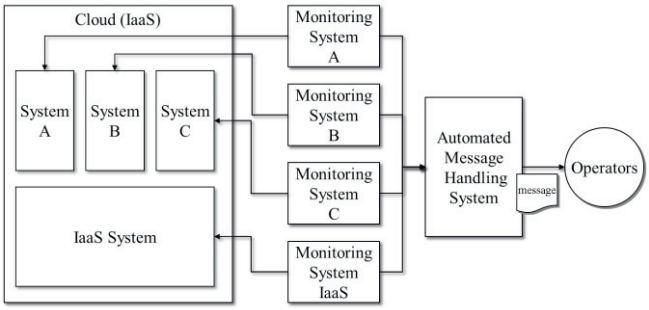

Figure 5 Overview of the data flow model

6. Evaluation Results and Discussion

6.1. Results

Table 5 represents the number of messages in the data center. More than 370 thousand messages are processed in 3 months by the prototype system.

Table 5. Number of messages

Month	Number of messages
December	86,924
January	177,035
February	106,221
Total	370,180

Runbook rules are applied to 295 thousand messages. The percentage of messages that apply to runbook rules is approximately 80% of all received messages. Table 6 lists the detailed number of messages. The timeline in Figure 6 represents the number of messages received and the number of messages processed by the runbook.

Table 6. Percentage of model applied to messages

Number of messages	370,180
Number of messages to which a runbook rule is applied	294,995
Ratio of messages applied to runbook rules	79.7%

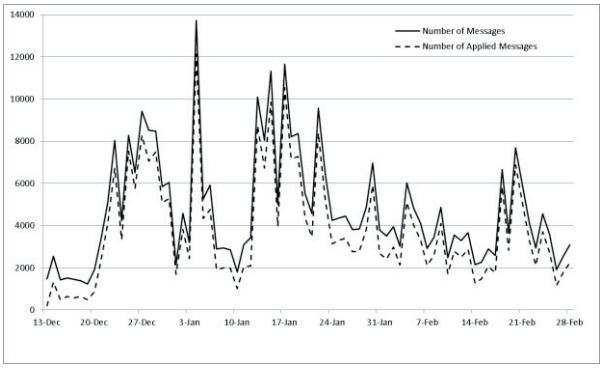

Figure 6. Number of message in chronological order

After the application of the runbook rules, approximately 40% of the messages are sent to the operators. Thus, 60% of the messages are deleted, and no human operator needs to take care of them. Table 7 lists the detailed number of results.

Table 7. Results of runbook rules

Results	Number of Messages	Percent
Send	117,706	39.9%
Delete	177,289	60.1%
Total	294,995	100.0%

6.2. Discussion

From our experiment in the data center, we obtained the following results.

- Runbook rule application ratio: 79.7%
- Ratio of deleted messages: 60.1%

From this result, we conclude that our runbook rules in the prototype system cover approximately 80% of the messages.

In this section, we estimate the time saved by the runbook rules. We assume that human operators spend 5 minutes on repeat rules. We also assume that they spend 5 minutes and 1 minute on a threshold rule and a message ignore rule, respectively. The total work time saved by runbook rules is shown in Table 8.

This result shows that 7,566 minutes per day was saved by runbook rules, which is 126 hours per day.

Table 8. Work time saved by runbook rules

Runbook rules	Operators' work time saved (min)	
	Total	per day
Repeat rules with timer and counter	310	4
Threshold rule	588,220	7,541
Message ignore rule	1,614	20
Total	590,144	7,565

We observed that a runbook rule was applied to 80% of the messages. We analyzed the remaining messages, which is approximately 20%. Many of the messages are used to check the monitoring systems themselves. We can add new runbook rules for these messages, although we thought that it is not suitable to apply rules to them.

Approximately 60% of the messages are deleted by the runbook rules. A total of 40% of the messages are sent to operators, which must be addressed by human operators. This procedure makes it easy for human operators to handle these messages because they already know that these messages should be addressed by humans.

The effect of work time analysis, which is described in Table 8, is based on the number of deleted messages. In practice, humans can handle several messages at once, and the effect on work time will be smaller than the effects described in Table 8.

7. Conclusions and Future Research

We proposed a state transition model for message handling in systems operation. Based on the model, we proposed an automated message handling system. The system helps humans to obtain an efficient operation. We built a prototype system and applied it to a commercial service. We observed that the runbook rules are applied to 80% of the messages. The runbook rules delete 60% of the applied messages. From the results, we estimated that 7,566 minutes of work time are saved per day.

The following three items are still open.

- Detailed analysis of messages to which the runbook rule is not applied. A new set of runbook rules should be designed to improve the efficiency of the operations.
- Automation of operators' actions, which are required after they receive messages.
- Design of intelligent runbook rules, which consider the network topology and the dependency of the sub-system and their components.

We observed that it is possible to apply the prototype system to other managed services. To apply the system, runbook rules are formally designed for the systems. When runbook rules are designed, they should not be designed based on the letters of system messages; instead, they should be based on the status of the systems. We must design the model of the system as state transition modes.

We are going to apply the concept of automated message handling system not only to the operations phase but also to the system design phase. This approach is now in the testing phase. The approach will be applied to more systems next year.

References

[1] Kerry Thompson: An Introduction to Logsurfer, SysAdmin Magazine March 2004, United Business Media plc, 2004
[2] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel, 2008, "The cost of a cloud: research problems in data center networks," SIGCOMM Compute, Communication, Rev. 39, 1 (December 2008), pp.68-73, DOI=10.1145/1496091.1496103
[3] Michael Isard, "Autopilot: automatic data center management," SIGOPS Oper, Syst, Rev. 41, 2 (April 2007), pp.60-67, DOI=10.1145/1243418.1243426
[4] M. Yoshino, M. Oba, N. Komoda, T. Yamade, and S. Nakamichi: , 2010, "Message Analysis Method Based on a Stream Database for Information System Management, " in Proc. of the 4th International Conference on Research Challenges in Information Science 2010 (RCIS2010), pp.519-526
[5] logsurfer (software) , 2004, http://www.cert.dfn.de/eng/logsurf/
[6] swatch (software), 2008, http://swatch.sourceforge.net/
[7] Hinemos (software), 2010, http://www.hinemos.info/
[8] Splunk (software), 2010, http://www.splunk.com/